JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 4, May-June 2008

Enabling Application Agility - Software as A Service, Cloud Computing and Dynamic Languages

Dave Thomas

BEYOND MIDDLEWARE

For the past decade, application developers have been forced into an increasingly complex labyrinth of multi-tiered hardware, complex OO frameworks and middleware and associated tools. Each year, developers have faced new frameworks and a sea of new APIs. Instead of making the life of application developers easier and reducing the cost of application development, these technologies have done just the opposite. We are only now beginning to understand the expense of the middleware legacy.

The good news is that application developers are on the verge of being liberated from the tyranny of middleware. Next Generation IT will leverage a new computing platform which makes the development and delivery of applications significantly easier than it is today. This new platform consists of Cloud Computing, Software As A Service and Dynamic Languages. Cloud Computing [1] offers mainframe or better infrastructure through a small set of services delivered globally over the Internet. Software as a Service is a new delivery model which provides flexibility to both the provider and the customers. Dynamic languages and modern frameworks lower the barrier for application development and enable the rapid development of applications.

SOFTWARE AS A SERVICE (SAAS) – ALWAYS UP TO DATE

The great benefit of SAAS is the ability, though hopefully not the requirement, to run the most recent version of the application. The pain of installation and upgrading is replaced by a simple request to run a specific version of the software. It eliminates the need to install hardware or software on the client premises.

A major challenge for SAAS providers is ensuring the security and privacy of client data which is held outside their organization. SSL VPNs and client based encryption as well as mission critical fault tolerant, environmentally and physically protected data centers are minimum capabilities to enable the use of external service

Dave Thomas, “Enabling Application Agility - Software as A Service, Cloud Computing and Dynamic Languages”, in *Journal of Object Technology*, vol. 7 no. 4 May-June 2008, pp. 29-32
http://www.jot.fm/issues/issue_2008_05/column3/

based software. SalesForce.com [4] is a pioneer in providing a robust secure infrastructure which has earned the trust of otherwise conservative corporations.

Currently, most applications require the user to be connected to the Internet to use their applications, but increasingly services are being designed such that they can work in an occasionally disconnected world which is the reality for mobile users. Google Gears [6], for example, provides a local web server which allows local data storage using simple web protocols.

SAAS enables the service provider to support many clients using a common infrastructure. It may even allow all clients to run a single or small number of instances of the application. Providers have the ability to collect detailed information about defects, performance and usage patterns to improve their product. Finally, SAAS allows updates, beta and new release features to be delivered to clients on a per user, team or company basis.

Google has clearly demonstrated the benefits of the software as a service model. Amazon Web Services [2] and recently Google App Engine [3] enable others to deliver their own applications leveraging their massive infrastructures.

CLOUD COMPUTING

Despite the massive investment in modern 2 and 3 Tier computing platforms, in many cases these platforms still fall short of mainframes when it comes to delivering enterprise applications in terms of scale and/or total cost of ownership. It isn't surprising then that IBM has announced a recent resurgence in mainframe sales. These systems require different kinds of programming for each tier. They require moving data to and from each tier.

Cloud Computing leverages high performance, relatively inexpensive commodity computers clustered closely together in a data center and connected to other similar data centers located globally as close to the users as possible. In contrast to middleware, Cloud Computing presents application developers with a small set of services (<100). These services provide a limited set of operations, reducing the learning curve for developers. Services [5] are invoked using simple REST XML requests and/or RSS, making them easy to use from any programming language. Convenience frameworks are provided for popular languages, most importantly low barrier dynamic languages.

While the initial service offerings were limited web and storage services, Amazon recently introduced their SimpleDB database services and Google's new APP Engine promises to allow developers access to its Big Table and Map Reduce infrastructure.

The latter allow a functional style of programming where functions are passed into the cloud infrastructure and executed close to the data rather than moving all of the data to the functions on a mainframe or mid-tier server. These higher level query and update services provide application developers the high performance data services of Cloud Computing via simple collection operations.

Legacy data can be made available through generic collection interfaces, reducing the complexity of middleware, persistence, and message bus APIs. We believe that we are now seeing the CRUD and the mainframe VSAM, IMS and RDB of the future.

Services can of course be composed such that every application is really nothing more than a composition of other services. Clouds enable loosely coupled services to execute efficiently inside the cloud eliminating or at least reducing the overheads of message buses, RPCs and persistence frameworks. Applications can be composed and orchestrated via simple end user tools such as mashup editors. The Programmable Web [5] documents the currently available Web APIs along with some interesting mashups.

DYNAMIC LANGUAGES

Dynamic languages such as Python, Ruby, Scheme, PHP, Java Script, VB9[1] and LUA etc. have much lower entry barriers than Java 6 and C# 3.0. The libraries are much smaller, they have a simple object model, and the objects rather than the variables are typed, increasing the polymorphism. This typically means less syntax and less code. This enables more agile development with easier refactoring and increased polymorphism[2].

Dynamic Languages also encourage the use of programming with collections and iterative style and most allow for functions to be easily passed as first class values supporting more a functional style of programming.

These languages encourage exploratory programming and can be used without the need for a complex IDE. Web application development is further enabled through frame works such as Ruby On Rails, and Django provides ready made scaffolding web sites which can built by application developers without deep technical knowledge of either Ruby, Python or the underlying web infrastructure.

Dynamic languages are ideal for writing small services which can be dynamically invoked on demand. The ability to leverage a powerful but very simple cloud service API reduces the complexity of persistence and messaging APIs, allowing the application to just deal with collections of data and invoke other services. This is in sharp contrast to J2EE where developers must weave/inject their application code into the middle of J2EE and deal with OR mapping, serialization etc. REST and RSS provide simple, ubiquitous protocols for a wide variety of communications. The complexity of CORBA and RPC and even SOAP is reduced through the use of JSON and YAML.

Ajax frameworks enable the development of rich client UI, or even complete applications which run in the browser scripted in JavaScript. The pervasive importance of such browser based application has resulting in browser vendors improving their browsers and scripting engines. While applications security remains a

[1] Local type inference and integrated support for LINQ have brought new life back to VB.
[2] Generics in C# and Java offer the opportunity for increased polymorphism but experience to date suggests that the complexity of generics and their interaction with other language features makes them less useful than designers had hoped.

concern, Google, Facebook, HP etc. have developed preprocessors that scan the JS code for potential problems and localize references to client data.

FINALLY LIFE AFTER MIDDLEWARE

Next Generation IT holds the promise that applications programming will no longer require climbing *API Mountain* and learning complex tools. This means increasing agility by significantly reducing application development time and expense. Instead of one monolithic application to deal with every type of customer or product, applications can be developed and delivered to types of customers, specific geographies or even to specific individual customers. Applications can instantly deploy on any Internet capable device.

Platforms such as Amazon, Google, iPhone, iPod, and Facebook, are used daily by millions of users. If your business wants to interact with these customers you need to reach them on their platform, not assume they will come to yours.

REFERENCES

[1] Nick Carr, The Big Switch: Rewiring the World, from Edison to Google, W. W. Norton.

[2] James Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB (Programming), O'Reilly.

[3] Google App Engine, http://code.google.com/appengine/ and Google Data APIs http://code.google.com/apis/gdata/

[4] Tony Stubblebine, An Introduction to Salesforce.com's AppExchange, http://www.oreillynet.com/pub/a/network/2006/11/13/an-introduction-to-saleforcecoms-appexchange.html

[5] The Programmable Web, http://www.programmableweb.com/

[6] Google Gears, http://code.google.com/apis/gears/

About the author

Dave Thomas is cofounder/chairman of Bedarra Research Labs (www.bedarra.com), www.Online-Learning.com and the Open Augment Consortium (www.openaugment.org) and a founding director of the Agile Alliance (www.agilealliance.com). He is an adjunct research professor at Carleton University, Canada and the University of Queensland, Australia. Dave is the founder and past CEO of Object Technology International (www.oti.com) creator of the Eclipse IDE Platform, IBM VisualAge for Smalltalk, for Java, and MicroEdition for embedded systems. Contact him at dave@bedarra.com or www.davethomas.net.